Subject: suggestion: code-parsing plugins
Posted by piotr5 on Mon, 13 Jul 2009 21:58:17 GMT
View Forum Message <> Reply to Message

Assist++ and syntax-highlighting both need to parse the code and sort of "understand" what is written. U++ adds to the standard c++ directives several new ones (i.e. uint32 and such) and they all happen to be coloured in a special way. my idea is that this particular colouring should be handled in a separate lib such that removing that lib from theide would also remove the syntax-highlighting for those. and I believe the same lib should also understand that CONSOLE_APP_MAIN macro is a function-definition for main(...) and whatever other macros U++ can offer, and this info should be available for Assist++. I imagine a very simplicistic plugin-system, where during compilation some script does just collect the names of all plugins into a dynamically created header-file and makes sure they get initialized at startup. i.e. assist++ would then just pass on interpretation to each plugin by calling the addresses stored in some container, which got filled during plugin-initialization. for example highlighting wouldn't need to use call-backs, the same for simple macros in Assist++. so it could still be a quite fast parsing. on the other hand a project which doesn't include Core could omit u++-plugin initialization and thereby speed up parsing.

what do you think? of course, as far as I can see no custom scripted file-creation seems to be implemented in theIde yet, so I cannot imagine how the plugin could get detected without manually altering the sources (beyond "ide.upp"). I call it plugin because I expect that merely adding the individual plugin-package to theide should be sufficient for it getting used. dynamically loaded plugins could also be a possibility, but statically linked plugins have the advantage that they get re-compiled everytime along with theide, and therefore don't require backwards-compatible interface. the idea is that users who never need u++ Core package can simply remove the overhead to make theide slightly smaller. and in general I would like to have theide more modularized such that unneeded features could just be excluded from compilation in future. but for now I expect that through such a plugin-system people could contribute their own special assist++ parser for certain libraries. and templates would also get a new use (especially since there would now be a code-parsing plugin-template). for example an stl-plugin could colour-code stl stuff, and it could even be kept up-to-date such that assist can understand whatever has been used in some stl-implementation...

oh, and I think that one needs to keep in mind that the parser of the configuration-file should take into account that not all plugins are loaded and thereby ignore the unused options. or maybe even name the config-file according to the loaded plugins resp. store plugin-related config into separate config-files.